Examen Parcial de Programación II – Ejercicio Práctico

1 de Junio de 2018

Realización: El ejercicio se realizará en la hoja de respuestas, que será lo único que se entregará. En ella se harán constar los apellidos y el nombre. Se pueden utilizar hojas de sucio aparte.
Puntuación: Este ejercicio tiene un peso de un 70% en la nota total del examen.
Duración: 1 hora y 15 minutos.
Calificaciones: Las notas se publicarán el día 11 de Junio.
Revisión: La revisión de este ejercicio tendrá lugar el día 14 de Junio a las 11h en el aula que se indicará en su momento.

Enunciado

Una *factoría* lleva un registro con todas las personas que acuden al trabajo a diario. De manera general, una *persona* se caracteriza por su *nombre* (String), su *domicilio* (String), su *DNI* (int) y su *hora de entrada* (*Horario*). La factoría admite a dos tipos de personas: los empleados y los visitantes. Los *empleados* son personas que tienen un *despacho* asignado (String) y un *número de registro de personal* (int); mientras que los *visitantes* son personas que llevan asociada una *clave* (int) que se obtendrá a partir de un *contacto* (String).

Para su control, la factoría lleva un registro diferenciado para cada tipo de persona, de tal manera que los empleados se registran automáticamente, mientras que los visitantes solo pueden registrarse si lo hacen a través de un empleado que se halle en el trabajo ese día; en ese caso el visitante asumirá como clave la propia del empleado en cuestión. Así mismo, la factoría podrá cancelar las visitas que lleven un determinado tiempo, para lo cual eliminará del registro de visitantes a los que se hayan pasado de ese plazo. Se supone que el orden de llegada de las personas a la factoría es cronológico.

En el diagrama UML inferior se hallan descritas las clases *Horario*, *Persona*, *Empleado*, *Visitante* y *Factoria*. Se pueden utilizar los TADs genéricos de lista y cola del curso. Se puede considerar hecha la implementación de la clase *Horario*. La especificación de los métodos de Horario, *esAnterior* y *diferenciaEnMinutos*, es la siguiente:

```
esAnterior (h : Horario) : boolean
POST: resultado es cierto si este horario es anterior cronológicamente a <h> y es falso e.o.c.
diferenciaEnMinutos (h : Horario) : int
PRE: <h> es anterior cronológicamente a este horario
POST: resultado es la diferencia en minutos entre este horario y <h>.
```

Se pide implementar los siguientes métodos de la clase *Factoria*:

Problema 1: (1 punto)

```
registrarEmpleado (e : Empleado)
EFECTO: Añade el empleado <e> a la factoría.
```

Problema 2: (2 puntos)

```
registrarVisitante (v : Visitante; contacto : String)
EFECTO: Añade el visitante <v> a la factoría asignándole como clave la del empleado cuyo nombre
es <contacto>.
```

Problema 3: (2 puntos)

```
registrarEntrada (p : Persona) throws AccesoDenegado
EFECTO: Añade la persona <p> a la factoría. Si <p> no es empleado o visitante se eleva una
excepción de tipo <AccesoDenegado>. (Nota: Utiliza los dos métodos anteriores)
```

Problema 4: (2 puntos)

```
balanceJornada () : String
EFECTO:  Construye un listado de tipo String con un par (<numeroRegistroPersonal>, <minutos>) para
cada empleado de la Factoria que no haya completado la jornada laboral.  <numeroRegistroPersonal>
representa el número de registro de personal del empleado y <minutos> es el número total de
minutos que le faltan para completar una jornada de 8 horas.
```

Problema 5: (3 puntos)

```
cancelarVisitantes (fin : Horario)
EFECTO:  Elimina de la factoría a todos los visitantes cuya hora de entrada sea anterior
cronológicamente a <fin>.
```

Diagrama

En el diagrama aparecen en **negrita** las operaciones que se piden.